Schrodinger's Snake - Project Guide

Joseph Paki

January 8, 2018

Contents

1 Introduction

1.1 Background

The Schrodinger's Snake project is a collection of Python assignments developed for the undergraduate quantum mechanics class at the University of Michigan. It is freely available on github, and for each assignment includes both a "problem notebook" (which you would give to students for homework) and a completed "solution notebook" (for the grader to check results and provide to students afterwards). The assignments take the form of Jupyter notebooks, which makes them very portable, allows students to both submit code and written responses to questions, and allows instructors to provide written feedback on assignments within the submitted notebooks. The general purpose of the project is two-fold:

1. Use the power and flexibility of Python / Jupyter notebooks to allow students to explore problems and concepts in quantum mechanics that are difficult / impossible to do via pencil and paper. In addition to allowing students to solve harder math problems, programming assignments can contain animations and plots that reveal behavior that is otherwise difficult to appreciate.

2. Expose physics undergraduates to programming tools and concepts. Given the wide range of jobs and research opportunities that require programming knowledge, properly equipping physics undergraduates for their future careers requires incorporating some level of programming experience into their training. Python is both a widely used and incredibly flexible programming language, as well as one of the easiest to learn. It is hoped that these resources will help students obtain a level of comfort with basic programming concepts, as well as empower them with another problem solving tool.

1.2 Getting Started

Working with the programs in Schrodinger's Snake requires Python (version 3.x is recommended) and Jupyter Notebook. The simplest way to get everything you need is to install Anaconda, a free and open source package of Python (and R) tools and libraries for scientific computing and data analysis. It can be downloaded at the Anaconda website, and I recommend getting the Python 3.x version to avoid issues with the differences between Python 2 and 3.

Once Anaconda is installed, you can start Jupyter by running the Anaconda Navigator. This visual interface is where you can launch various tools (like Jupyter), and also where you can update and install Python libraries. A panel with the Jupyter Notebook logo should appear on the front screen - click on the Launch button below it, and Jupyter will open up in a new browser tab.

2 Guide to the Assignments

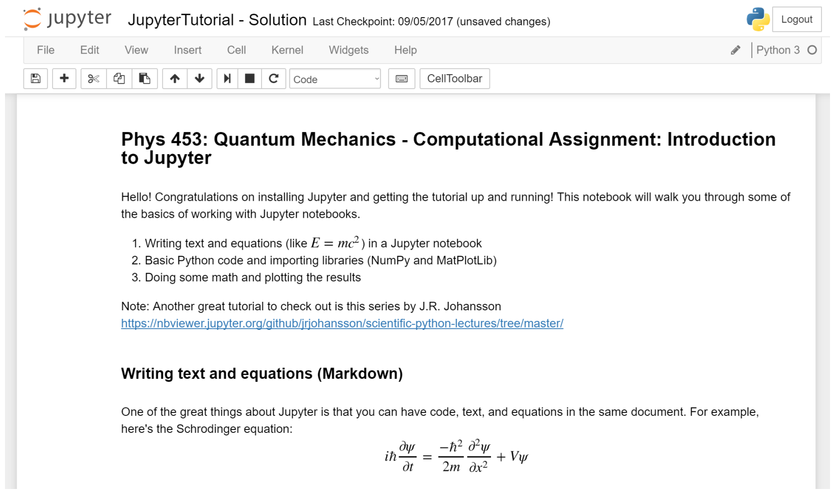
The following is a summary of the topics and programming concepts covered by the various notebooks in Schrodinger's Snake. They are listed roughly in the order in which topics might come up during an undergraduate quantum mechanics class. The individual projects are contained in folders (of the same name) in the GitHub repo.

2.1 Introduction to Jupyter

This provides:

1. a PDF the guides students on how to install Anaconda and how to run Jupyter, as well as provides some context for why scientific computing is a critical component of modern science.
2. A notebook that walks students through some of the basics of Python and how to work with Jupyter

This is typically the first computational assignment given to students, and is intended to get everyone (even those with no programming experience) up to speed quickly and prepare them for the following assignments.



Phys 453: Quantum Mechanics - Computational Assignment: Introduction to Jupyter

Hello! Congratulations on installing Jupyter and getting the tutorial up and running! This notebook will walk you through some of the basics of working with Jupyter notebooks.

1. Writing text and equations (like $E = mc^2$) in a Jupyter notebook
2. Basic Python code and importing libraries (NumPy and MatPlotLib)
3. Doing some math and plotting the results

Note: Another great tutorial to check out is this series by J.R. Johansson https://nbviewer.jupyter.org/github/jrjohansson/scientific-python-lectures/tree/master/

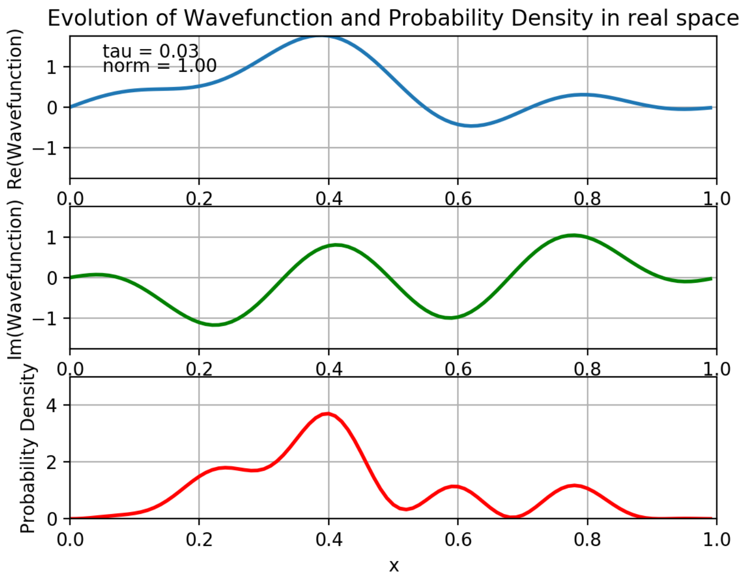
Writing text and equations (Markdown)

One of the great things about Jupyter is that you can have code, text, and equations in the same document. For example, here's the Schrodinger equation:

$$i\hbar\frac{\partial\psi}{\partial t} = \frac{-\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} + V\psi$$

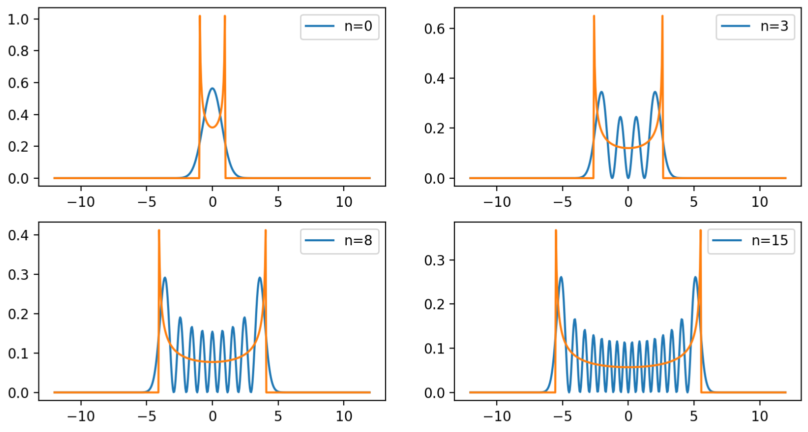
2.2 Infinite Square Well

This assignment explores the behavior of stationary states and superposition states. Students learn how to represent wavefunctions and space with discrete arrays, and they produce animations of that allow them to compare the time-depedence of stationary states and arbitrary superpositions of energy eigenstates. The end of the notebook also introduces them to momentum space.



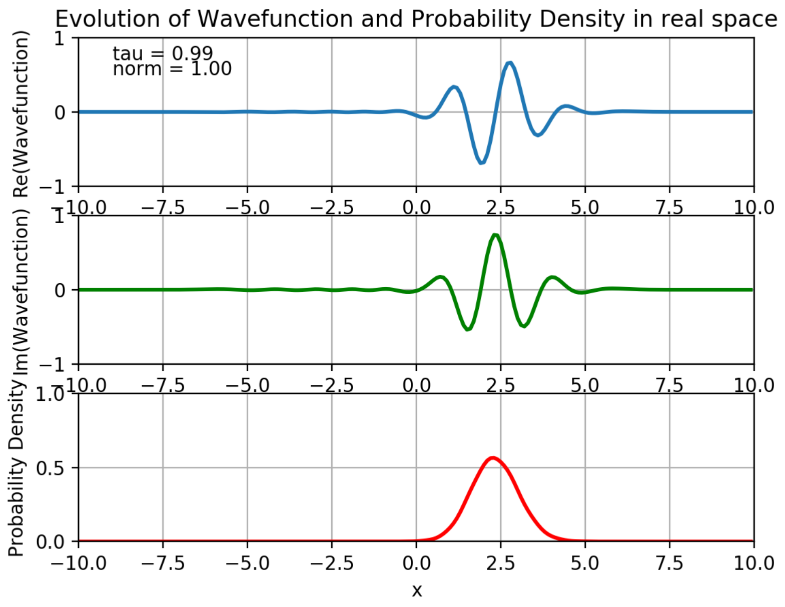
2.3 Simple Harmonic Oscillator - Plotting Eigenstates

Compares the classical SHO to the quantum SHO. Students program the energy eigenstate wavefunctions of the SHO and compare the classical probability distribution to that obtained from $|\psi_n(x)|^2$, and take note of the comparison as $n \to \infty$. This also introduces them to the special functions available through SciPy, as needed for the Hermite polynomials.



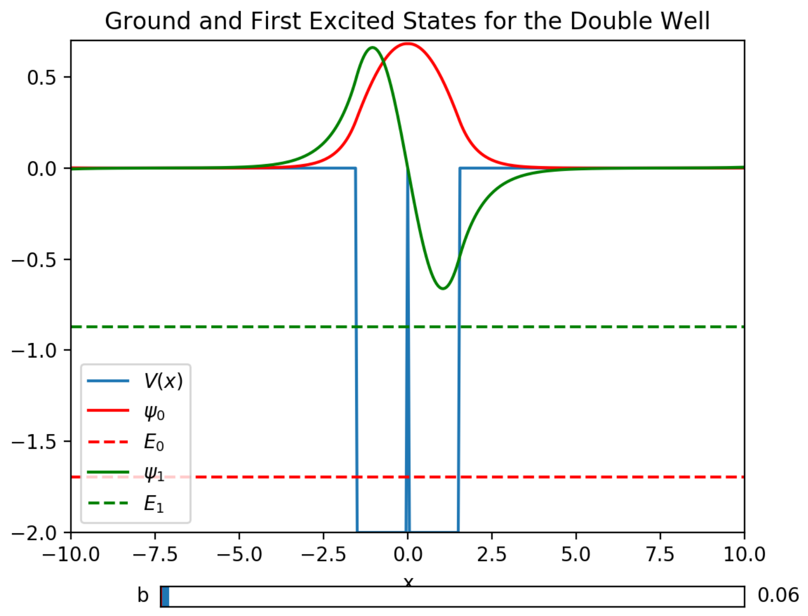
2.4 Simple Harmonic Oscillator - Coherent States

Students learn how to represent and build SHO coherent states via superpositions of energy eigenstates. Animations allow them to compare the time-dependence of coherent states with arbitrary superpositions. They also consider the consquences of truncating the infinite series that defines the coherent states.



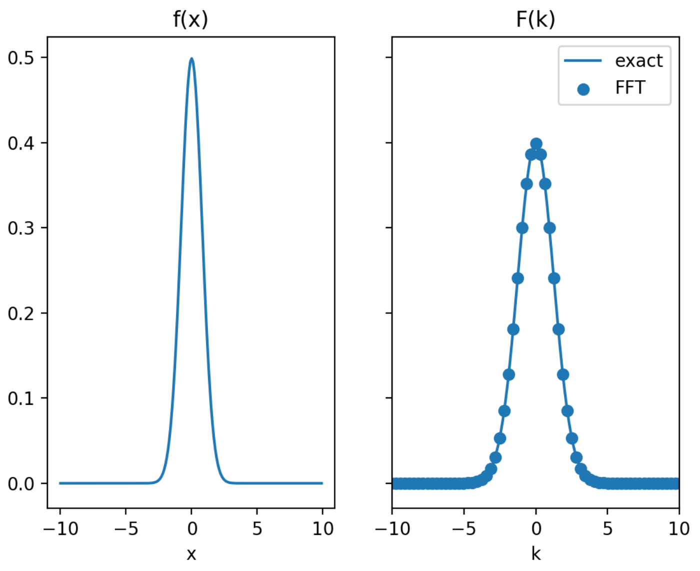
2.5 Double Finite Square Well

This notebook is meant as a demonstration of the shooting method for numerically solving for the eigenenergies and eigenstates of the double finite square well. It includes an interactive plot that allows students to see how the energies change as the distance between the two wells changes.



2.6 Fourier Transform

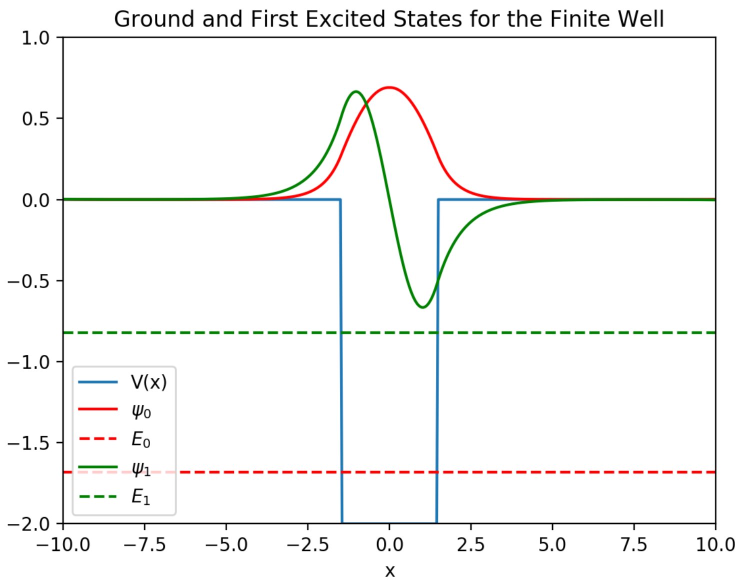
This assignment walks students through how to perform Fourier transforms with NumPy. As the notebook describes, this is not entirely straightforward since the continuous Fourier transform must mapped onto the discrete Fourier transforms supported by the NumPy library.



2.7 Finite Square Well + Transfer Matrix

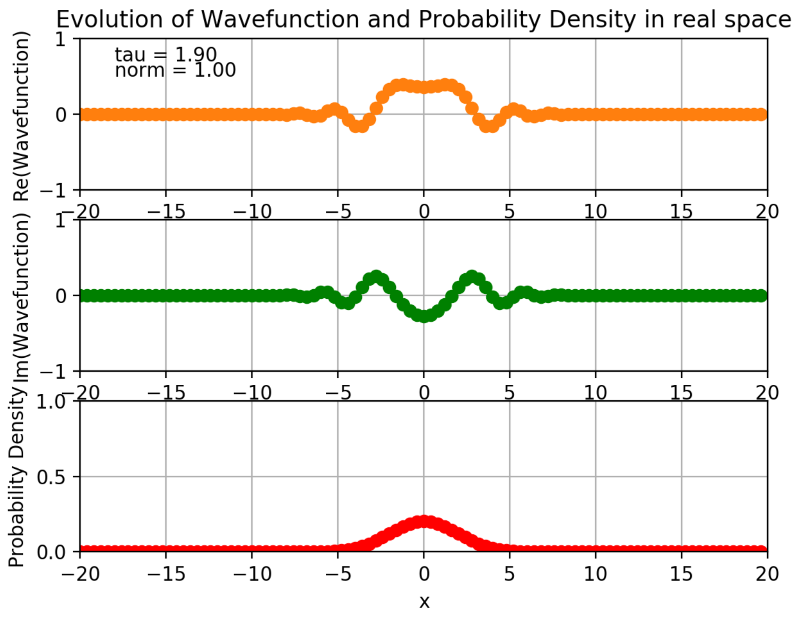
Student use the shooting method to numerical solve for the first two energy states of the finite square well. They also calculate expectation values and use Fourier transforms to verify the space-momentum uncertainty principle. As an extra credit challenge, they can also try to use transfer matrices to find the transmission coefficient through two finite square wells as a function of energy. (This notebook assumes students are familiar with the Shooting Method and Fourier Transforms.

```
Norm Psi_0= 1.0
Energy_0= -1.6805031159892678
Norm Psi_1= 1.0
Energy_1= -0.8209590911865234
```



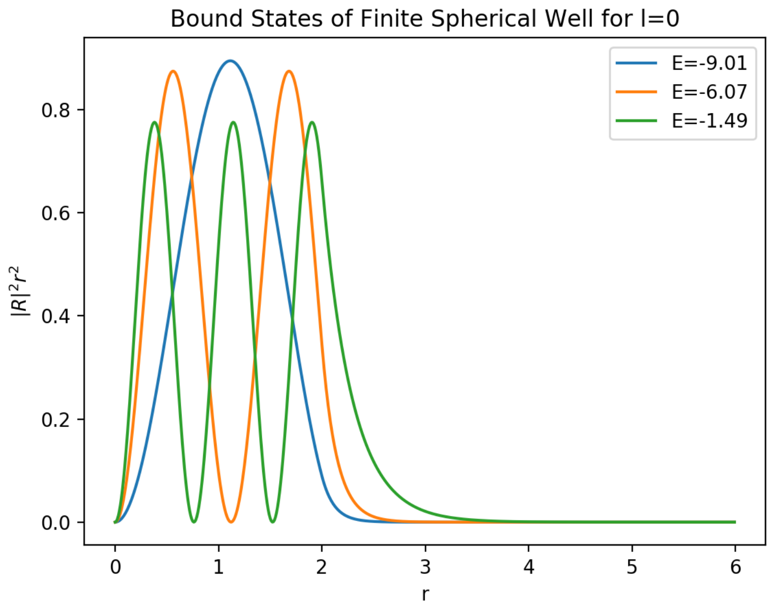
2.8 Gaussian Wavepacket

Demonstrates how one can solve for the time evolution of a Gaussian wavepacket by Fourier transforming into momentum space, applying the correct time evolution to each momentum mode (free particle state), and then transforming back into real space.



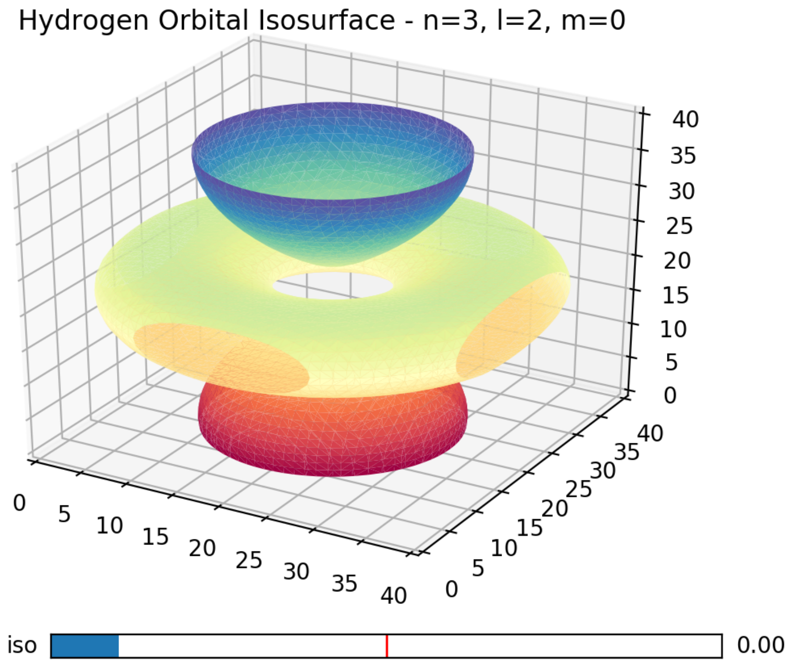
2.9 Finite Spherical Well

Students learn how to numerically obtain the bound states and energies for the finite spherical well. This involves using the special functions library from SciPy, as well as using root finding to find the erergies that solve the transcendental equation that arises from enforcing boundary conditions.



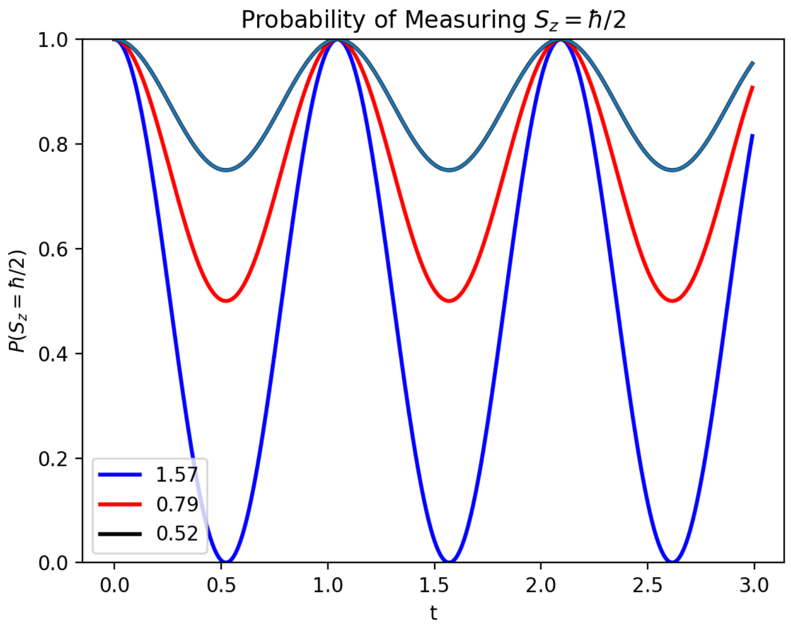
2.10 Hydrogen

This notebook is meant for demonstration purposes. It forms the 3D wavefunctions of the hydrogen atom and visualizes them in a number of ways, including 2D heat maps and a 3D isosurface plot.



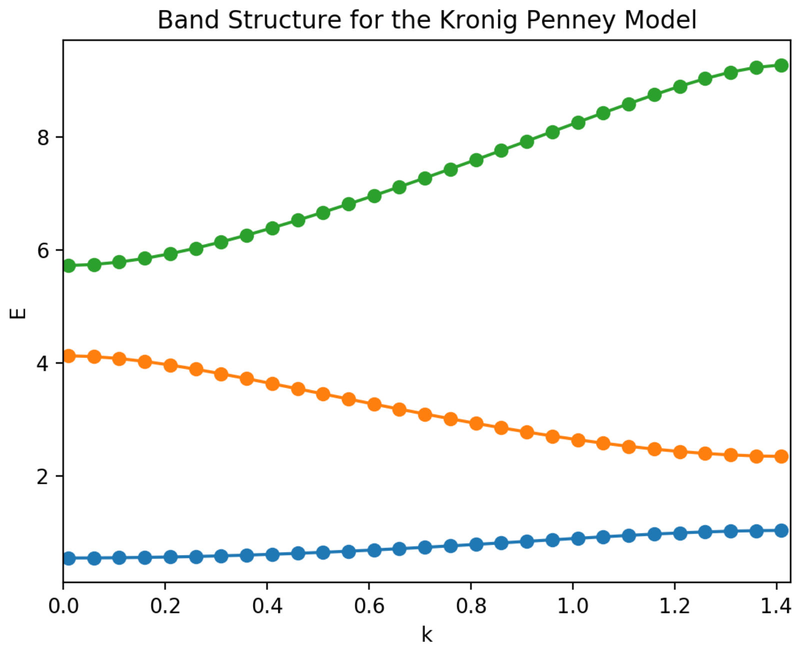
2.11 Spin Precession

Students numerically solve the problem of a spin 1/2 particle in a uniform magnetic field. After coding up the Pauli matrices and Hamiltonian, they find the eigenvectors and energies and use them to transform the problem into the eigenbasis. They then find the time evolution of the probability of measuring the particle to be spin up in the z direction for various directions of the magnetic field.



2.12 Kronig Penney

Students first find the set of four equations that arise from enforcing boundaries conditions and Bloch's theorem on the Kronig Penney Model (repeating finite square wells). They then numerically find the energies that solve this system (ie, the energies that make the determinant of the system matrix equal to zero) as a function of the wavevector. This allows them to find the full bandstructure and density of states of the model. The written response questions also introduce them to solid state concepts such as band gaps and band transitions.


Band Structure for the Kronig Penney Model
E
k